Naval Research Laboratory
Washington, DC 20375-5320

NRL/MR/6180--05-8882

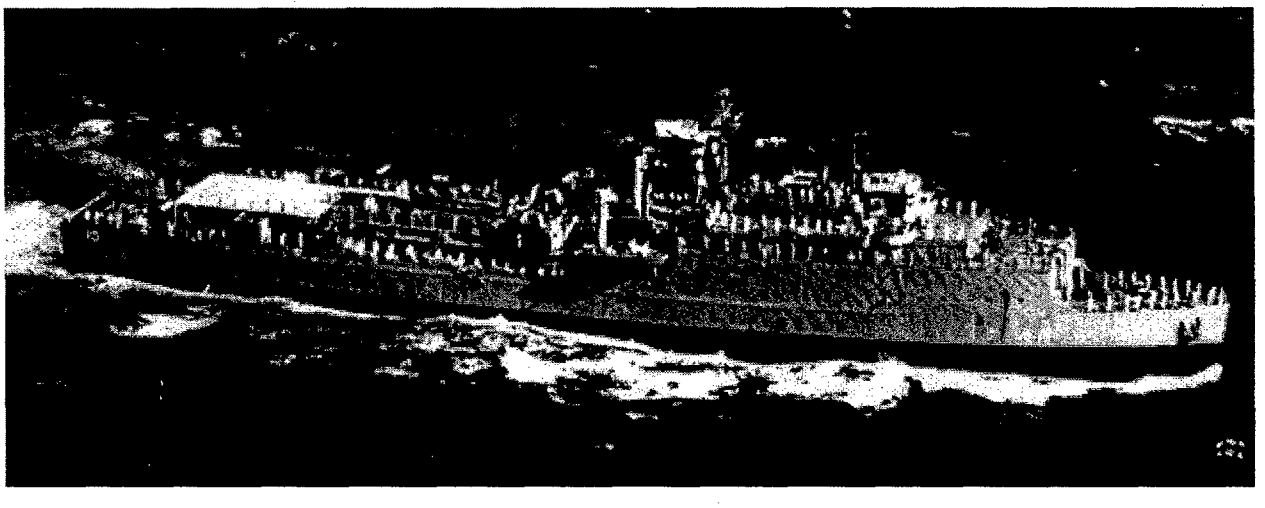
Creating a Digital Ship Model from AutoCAD Drawings

TOMASZ HAUPT
GREGORY HENLEY
Center for Advanced Vehicular Systems
ERC, Mississippi State University
Starkville, MS

PATRICIA A. TATEM
ITT Industries-Advanced Engineering Sciences
Alexandria, VA

FREDERICK W. WILLIAMS
Navy Technology Center for Safety and Survivability
Chemistry Division

June 8, 2005



REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
06-08-2005	Memorandum Report	October 2004-February 2005

4. TITLE AND SUBTITLE

Creating a Digital Ship Model from AutoCAD Drawings

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Tomasz Haupt,* Gregory Henley,* Patricia A. Tatem,† and Frederick W. Williams

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Research Laboratory, Code 6180
4555 Overlook Avenue, SW
Washington, DC 20375-5320

8. PERFORMING ORGANIZATION REPORT NUMBER

NRL/MR/6180--05-8882

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Naval Surface Warfare Center, Carderock Division
9500 MacArthur Blvd.
West Bethesda, MD 20817-5700

10. SPONSOR / MONITOR'S ACRONYM(S)

11. SPONSOR / MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

*Center for Advanced Vehicular Systems, Mississippi State University, Starkville, MS
†ITT Industries-Advanced Engineering Sciences, Alexandria, VA

14. ABSTRACT

Numerical modeling is emerging as a critical element of the ship design process. Automatic conversion of CAD into digital geometries is opening vast opportunities for rapid numerical modeling to evaluate ship designs and design philosophies and to quickly optimize design concepts to meet required performance and recoverability goals. As the ship concept is refined, simulations can continue to evaluate ship vulnerability and to begin the processs of defining ship operations. This work demonstrates the feasibility of the automatic conversion of the CAD drawing into the model input files.

15. SUBJECT TERMS

Network model; CAD; Fire model; Relational database

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Frederick W. Williams
Unclassified	Unclassified	Unclassified	UL	26	19b. TELEPHONE NUMBER *(include area code)* (202) 767-2002

CONTENTS

Executive Summary

Within this project, parts of the CVN-21 representing two partially overlapping areas have been successfully reconstructed. The AutoCAD files have been converted into a digital representation of the ship and stored in a relational database. The digital representation of the ship has been used for generating a three-dimensional visual model of the ship and serves as a Graphical User Interface (GUI) for the Fire and Smoke Simulator (FSSIM) for visualizations of the simulation results. Most importantly, we used the information stored in the database for the automatic generation of the input files for the FSSIM.

Numerical modeling is emerging as a critical element of the ship design process. Automatic conversion of CAD into digital geometries is opening vast opportunities for rapid numerical modeling to evaluate ship designs and design philosophies and to quickly optimize design concepts to meet required performance and recoverability goals. As the ship concept is refined, simulations can continue to evaluate ship vulnerability and to begin the process of defining ship operations. This work demonstrates the feasibility of the automatic conversion of the CAD drawing into the model input files.

In spite of the phenomenal rate of advancement in hardware and software, computers cannot compete with the human eye and brain in recognizing and understanding complex drawings. Therefore, in order to process the CAD files programmatically, the drawings must conform to certain rules that simplify the analysis. Certainly there is a cost associated with converting the existing drawings to such standards and to improving the shipyard drawing practices. There are enormous benefits of digital processing that would likely outweigh the effort to meet these standards.

Most of the algorithms to construct the digital representation of the ship come from the previous work with a small test area of ex-USS Shadwell and ex-USS Peterson. Because of its size, the CVN-21 project proved to be far more complex and revealed some deficiencies of our software capabilities that needed to be analyzed and corrected. However, the most laborious part of this effort was encountered with drafting errors, ambiguities and inaccuracies that result from the drawing practices adopted. These issues are not obvious when viewing the drawings. In fact, most are transparent to the observer. For the most persistent problems, such as inaccuracies of the positioning of the ship elements, new complex algorithms have been developed. Similarly, there was a need to develop additional procedures for recognition of the often inconsistent drawing patterns for representing doors, portholes and archways, instead of straightforward processing of AutoCAD blocks, as was the case for ex-USS Peterson (another inefficiency of the CVN-21 AutoCAD drawings). Finally, to handle numerous exceptions, ambiguities and complicated patterns, the files obtained from the shipyard had to be manually corrected. This manual drawing procedure was very laborious and error prone, making the processing difficult. Ideally, these corrections should be performed by the design group so problems are fixed for future revisions rather than repeating the same corrections for each new revision.

Another important area to significantly improve the current drawing standards in order to make them suitable for the digital processing is the handling of object properties (user-defined attributes attached to block references, for example). This is of critical importance for vertical

junctions, such as doors. The CVN-21 AutoCAD drawings are two-dimensional. There is no way of capturing the variable height of the doors or the presence and size of the sill short of inserting explanatory text. Taking into account the inconsistency of inserting the text into the CVN-21 files, a programmatic identification of door sizes is much more difficult to use by the designer than is a straightforward reading of the block properties.

To take full advantage of numerical simulations as a design optimization tool, it is necessary to automate the process of converting the CAD drawings into the models' input files. This work demonstrates that it is feasible, if the drawings are prepared with digital processing in mind. That is, if standards are followed, certain rules would make the drawings unambiguous and accurate. To achieve that end, it is necessary to employ procedures and software tools to assess the conformance of the drawing to such standards. The result of using such procedures and software tools could possibly increase productivity of the design personnel and increase the quality of the drawings, sufficient for a rapid, programmatic extraction of the ship geometry to become the input for numerical simulations of various aspects of the ship design.

CREATING A DIGITAL SHIP MODEL FROM AUTOCAD DRAWINGS

1. OBJECTIVES

Numerical modeling can potentially become a critical element in ship design, in particular, when used to evaluate ship designs and design philosophies as an overall conceptual design to quickly meet required performance goals. As the ship concept is refined, simulations can be used to evaluate ship vulnerability and recoverability and to begin the process of defining ship operations.

A starting point for performing simulations of a ship is defining the geometry. In the current project, the geometry has been provided by the vendor as a set of AutoCAD [1] files, in the native drawing file (DWG) format. Each file represents one or more decks as a two-dimensional deck plan. The purpose of this project is conversion of those files into input data sets for the network fire model Fire and Smoke Simulator (FSSIM) [2].

Basically, FSSIM must be provided with a list of compartments defined as closed volume constructed by sides (bulkheads, overheads and decks) and a list of junctions (e.g., doors, archway, and hatches). Each element in the lists must be described by a set of parameters that control evolution and spread of a fire. For example, each side is characterized by an area, thickness and composition, information that is used to compute the heat flow from one compartment to another. Similarly, parameters describing air flow through the junctions must be supplied. The FSSIM model requires the input data to be formatted as a list of FORTRAN NAMELIST statements. The complete definition of the input file format can be found in the FSSIM manual [3].

For a large ship, such as the CVN-21 aircraft carrier, which has hundreds of compartments and junctions, manual creation of the model input data is unreasonable, unless the process is at least partially automated. The objective of this project is, thus, the design and prototype implementation of software tools for the automated creation of a digital, three-dimensional model of the ship from the AutoCAD files. This project creates new opportunities for developing new simulation-driven design, test, validation and verification tools that will facilitate the ship design process.

2. GENERAL APPOACH

Processing of CVN-21 AutoCAD files is performed in four steps:

1. **Conversion to DXF format.** Unlike DWG, Data Exchange Format (DXF) is an open text format. The format is open because the specification is free and publicly available from AutoDesk [4]. Since DXF is a text (ASCII) format, the DXF files are portable between various computer platforms and easy to parse. The conversion between DWG and DXF format is provided by AutoCAD software under the "Save As" option.

Manuscript approved March 28, 2005.

2. **Parsing DXF.** A parser in Java has been developed so that it interprets DXF input, line by line, and creates Java data structures representing the "raw" AutoCAD data in computer memory.
3. **Analysis of the parsed data.** The next step is the processing of the parsed data to identify compartments, junctions and their parameters. The details of this process are described in the following sections of this document. The results of this processing are stored in a relational database. The design and the schema of the database are described in our earlier report [5]. The advantage of storing the processed data in a relational database is the clear separation of two independent processes: analysis of the AutoCAD data and generation of model-specific data in model-specific data formats. In particular, it is possible to produce input data for different models in different formats without reparsing and reanalyzing the AutoCAD input. The data for different models can also be generated in parallel, if, e.g., two different models are to be run concurrently. In our earlier work [5], the information stored in the database had been used to generate input data for the FSSIM model, creating a graphical three-dimensional ship model and visualizing the simulation results in real-time. Also, using the database allows one to modify the original geometry for specific model requirements, for example, to simulate bulkheads destroyed by a blast.
4. **Generation of the model-specific input data sets.** This process involves database queries that format the results based on the model specification.

Step 1 does not require any effort on our part. The format conversion is supported by tools embedded with the AutoCAD package. Steps 2 and 4 are relatively easy and do not involve any research issues and thus, represent a solvable software engineering problem. The rest of this report is, therefore, devoted exclusively to the analysis of the parsed AutoCAD data.

3. CHARACTERISTICS OF THE AUTOCAD DATA

AutoCAD data formats (DWG, DXF, and others) are essentially graphical formats. This means that the information in the AutoCAD files comprise only graphical elements, such as line segments, polylines, texts or curves, and not objects that are drawn using these graphical elements, such as compartments, doors or scuttles, or a description of a particular object. In particular, parsing a DXF file results in lists of graphical elements used in the drawing, e.g., a list of line segments or a list of texts. Each element of a list is distinguished by its coordinates. For example, a line segment is defined by the coordinates of its end points. The goal of the analysis of the parsed data is, thus, recognition of the objects or, in other words, selecting a set of graphical elements that represents, e.g., an archway, as shown in Figure 1.

Figure 1: *A graphical representation of an archway. The archway is drawn in CVN-21 AutoCAD as a closed polyline (a rectangle) with a dashed line that connects bulkheads on both sides of the archway.*

Each graphical element may have attributes assigned to it. For example, the set of attributes for a line segment include width of line, color, and type (continuous or dashed). Some attributes are defined by the user, e.g., phantom line type. These attributes provide means to distinguish between different line segments, in addition to the coordinates of its vertices.

AutoCAD also allows graphical elements to be grouped into so-called blocks. There are two major advantages of using blocks. First, once the block is defined, it can be reused as often as needed. For example, there are decks in CVN-21 with hundreds of archways. It is sufficient to draw its graphical representation (such as that shown in Fig. 1) only once and then insert copies into desired locations. In such a case, the insertion point of the block is listed in the DXF file, an easy target for our DXF parser. Second, arbitrary user-defined attributes can be associated with a block. An attribute can be used for specifying the object properties which are otherwise impossible to specify in a 2-dimensional drawing short of inserting a label, such as door height and the existence of a sill. The attributes can specify also some other properties of an object, such as composition or security requirements.

Finally, AutoCAD allows organizing the graphical elements into layers. One benefit of using layers is to provide yet another method of distinguishing between different instances of the same graphical element type. For example, each compartment is described textually in two ways: by its name (e.g., "CREW LIVING SPACE") and its locator encoded by deck-frame-position-use type designator (e.g., "3-56-4-L"). In DXF format, both are just text elements. To differentiate in the analyzing program, the text would have to be parsed and some knowledge used to assign the text into one of the two categories. Drawing these two texts in two different layers trivially solves the problem. In DXF format, elements of each layer are listed separately.

4. RECONSTRUCTION OF COMPARTMENTS

4.1 Overview

A compartment is a closed volume bounded by sides (bulkheads), a deck and an overhead. In terms of AutoCAD graphical elements, it is a closed shape (typically a rectangle, but in many cases the shape of the compartment is much more complicated.). To reconstruct a compartment we need to:

1. Select line segments representing bulkheads
2. Add the third dimension, assuming that all bulkheads are vertical and form a right angle with respect to the deck surface and have a constant deck height
3. Add the deck defined as a polygon formed by line segments, representing the bulkheads
4. Add the overhead by copying the deck and positioning it as parallel to the deck and moving it up by the height of the deck. Effectively, that lays the overhead on the surface of the deck, one level above the current one, if any
5. Select the text describing the name of the compartment
6. Select the text describing the location of the compartment
7. Compute the compartment parameters, such as volume

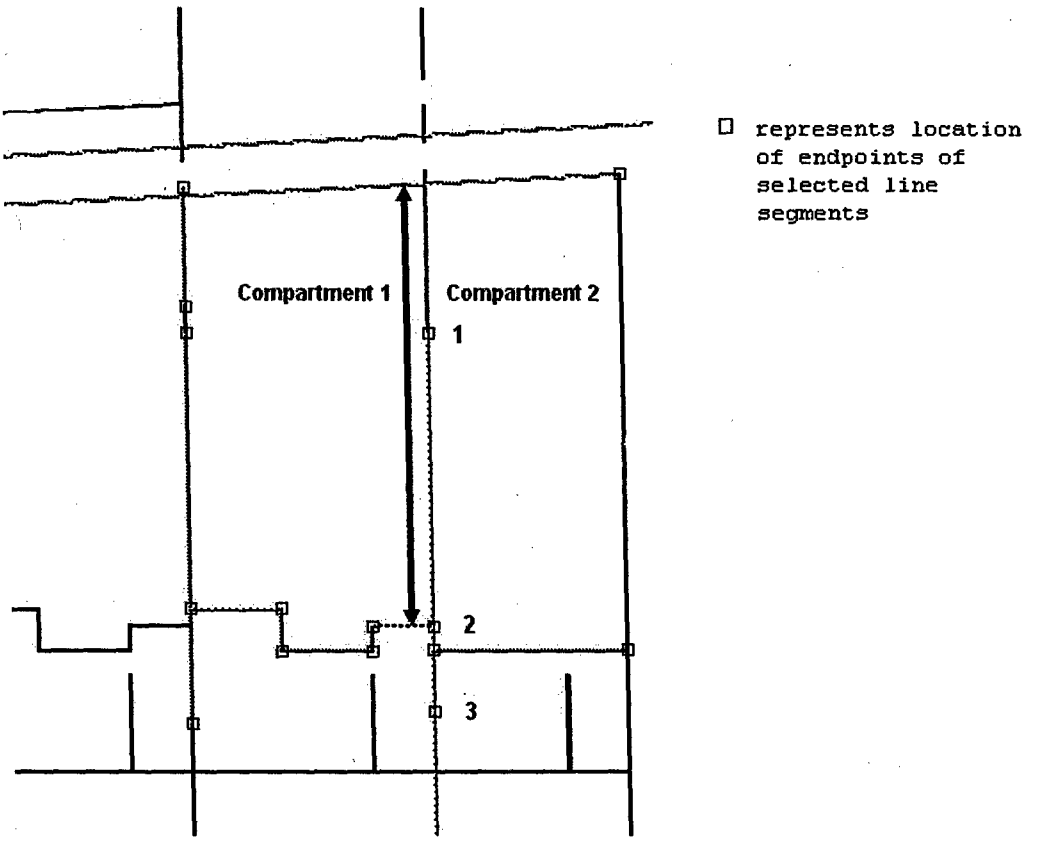


***Figure 2:** A fragment of AutoCAD drawing showing a deck.*

In most cases, a line segment does not directly correspond to a bulkhead; it is either longer or shorter than the actual bulkhead. As discussed in the Objectives above, the bulkhead represents a surface through which the heat transfers from one compartment to another. Therefore, a bulkhead must be uniquely identified by exactly two compartment identifications. As shown in Fig 2, the CVN-21 is not drawn this way.

Small rectangles in Fig. 2 show the location of endpoints of the selected line segments. Let's concentrate on a line separating Compartment 1 and Compartment 2. The double-headed arrow superimposed on the CAD drawing in Fig. 2 indicates the extent of the actual bulkhead separating these compartments. However, the CAD drawing is done differently. Instead, we see a line segment, starting at node 1, going upward beyond the compartment boundaries, across a narrow passage, and further. Then, there is another line segment connecting node 1 and 3 (node 2 is the endpoint of a horizontal segment marked with a bold dotted line). To form the bulkhead, it is necessary to break the first line segment at the end of the compartment, break the second segment at node 2, and combine the two broken segments into a single one. Note also that the first segment when crossing the passage is not continuous. The apparent hole coincides with an archway (the layer where the archway is actually drawn is switched off, for the sake of clarity in Fig. 2).

An analogous operation must be performed when reconstructing decks and overheads, since the deck plans of any two adjacent decks do not match. Therefore, it is necessary to

superimpose the two deck plans and break the lines in order to reconstruct common surfaces between pairs of compartments. Consequently, in most cases, a compartment deck and overhead is composed of several pieces.

4.2 Selecting Lines Representing Bulkheads

The majority of bulkheads are drawn on a layer labeled BHDS as polylines with a specific line weight (bold). This greatly simplifies the process of selecting relevant CAD graphical elements for further analysis. This is a different convention as compared to that employed in our previous test case, the ex-USS Peterson).

This is a very useful convention, but is not always observed. There are several types of issues that have been observed in the AutoCAD files:

- Bulkheads drawn as lines instead of polylines (which makes it impossible to distinguish them from other elements also drawn as lines in this layer)
- Bulkheads drawn as thin polylines (which makes it impossible to distinguish them from other elements also drawn as thin polylines in this layer)
- Bulkheads drawn in other layers (which makes it impossible to find them automatically, since polylines in other layers do not represent bulkheads)
- Bulkheads deliberately drawn in other layers (for example, all recent modifications or revisions are drawn in separate layers, which is very handy when comparing different versions of the ship design. The very fact that the bulkheads are drawn in several layers does not pose any difficulties for automatic parsing and analysis. The problem with the revision layers is that they contain all objects added or modified within the revision. This makes it difficult to distinguish between polylines that represents bulkheads and polylines representing other objects)

A failure to identify a bulkhead element results in the failure of the compartment recognition algorithm, since we are not able to create a closed polygon. We have not identified any practical way of protecting against this. We identify missing (i.e., unidentified) bulkheads only after attempting the compartment reconstruction. Since this is not the only reason for which the reconstruction might fail and taking into account the size and complexity of CVN-21, identification of the problem is tedious and labor intensive, in spite of the tools that we have developed to date. We correct the misidentification problems by manually correcting the CAD drawings, for example, by moving an object from one layer to another.

Many polylines representing the bulkheads are two-node polylines, thus, graphically indistinguishable from simple line segments. Nevertheless, it is important that they are represented as polylines which simplifies the selection of bulkhead elements. The remaining bulkheads are represented as "true" multi-node polylines which, in some cases, approximate curved surfaces.

4.3 Removing Zero-Length Segments

Unfortunately, some of those polylines are drawn by mistake. They comprise segments of zero length which certainly do not represent actual bulkheads. We handle such cases by automatic removal of the zero-length segments; and if such a segment happens to be between non-zero-length ones, we automatically reassemble the polyline. This operation is performed after parsing, so that the original drawings are not affected. The zero-length segments are not visible on the computer screen; however, we can provide a tool for identifying them or provide a list of them.

4.4 Breaking Polylines Into Line Segments

Line segments are much easier to process than polylines. A line segment is represented as a pair of nodes, while a polyline is a variable-length list of nodes. Breaking polylines is a relative straightforward operation, done in the computer memory; thus, the original CAD drawings are not affected. From this moment on, bulkhead elements are represented as simple line segments, regardless of how they are represented in CAD. Therefore, the same algorithms can be used for processing the CVN-21 drawing and ships drawn using different conventions, such as ex-USS Peterson.

4.5 Conditioning Line Segments

4.5.1 Overview

The line segments obtained by breaking polylines, in most cases, do not represent the actual bulkheads. Some of them are drawn inaccurately (Section 4.5.2); some of them are drawn in error (Section 4.5.3); and some of them represent a part of a bulkhead or multiple bulkheads (Section 4.5.4).

4.5.2 Accuracy of CVN-21 AutoCAD Drawings

The objects in the CVN-21 CAD drawings have created some issues, which make automatic analysis of the drawings very complex and sometimes ambiguous. One of these issues is that the endpoints of the adjacent line segments do not coincide, even though they logically should (e.g., the two line segments form a corner of a compartment). A few typical cases are schematically shown in Fig. 3.

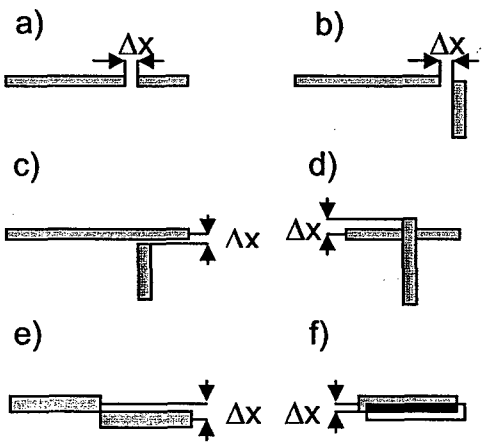


Figure 3: *Issues in CVN-21 CAD drawings. (a,b): Δx is the distance between two points (the endpoints of the adjacent bulkheads). (c,d): Δx is the distance between a line and a point (a bulkhead and the endpoint of another bulkhead forming a tee). (e,f): Δx is the distance between two lines (two bulkheads). For a precise drawing, we expect Δx=0 in all cases. For the sake of clarity, in this drawing we show the distance between the objects along only one axis. In the actual code we compute the distance in two dimensions.*

The information in Fig.4 shows the distribution of Δx; that is, the actual distances of the points that are expected to coincide.

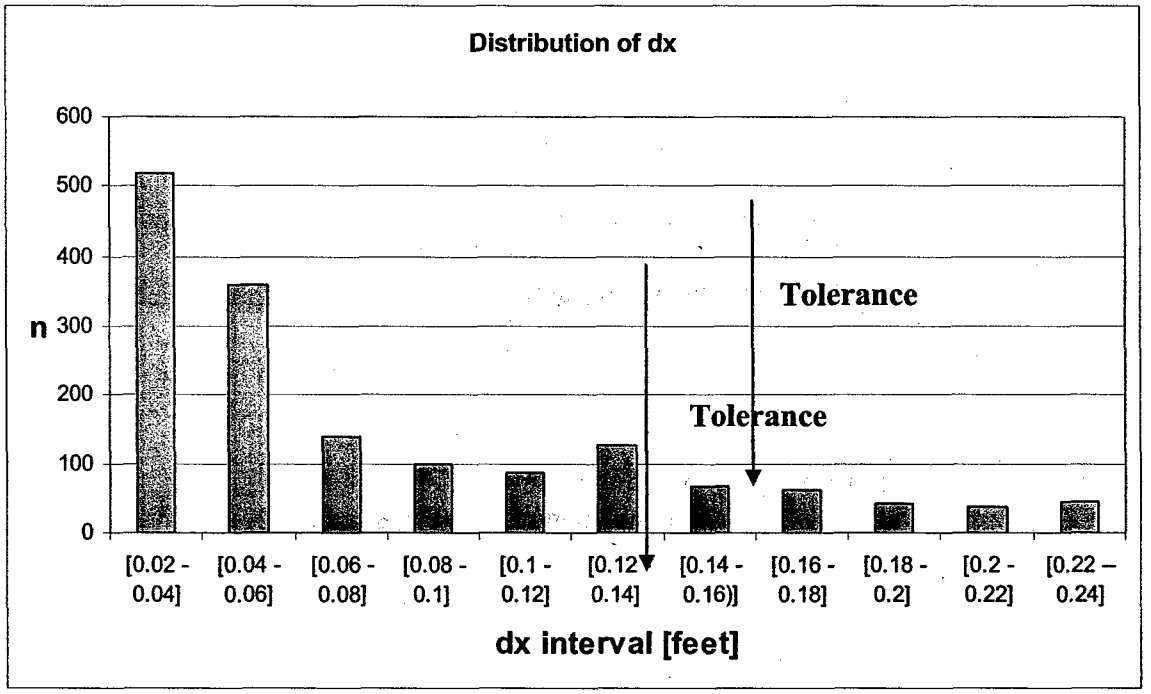


Figure 4a: *Distribution of dx (Δx as defined in Fig. 3a, b) for all decks of CVN-21. Most of the pairs of points have dx<0.02 ft (not shown); the displayed sample of the pairs constitutes about 0.1% of all pairs with dx<0.02 ft.*

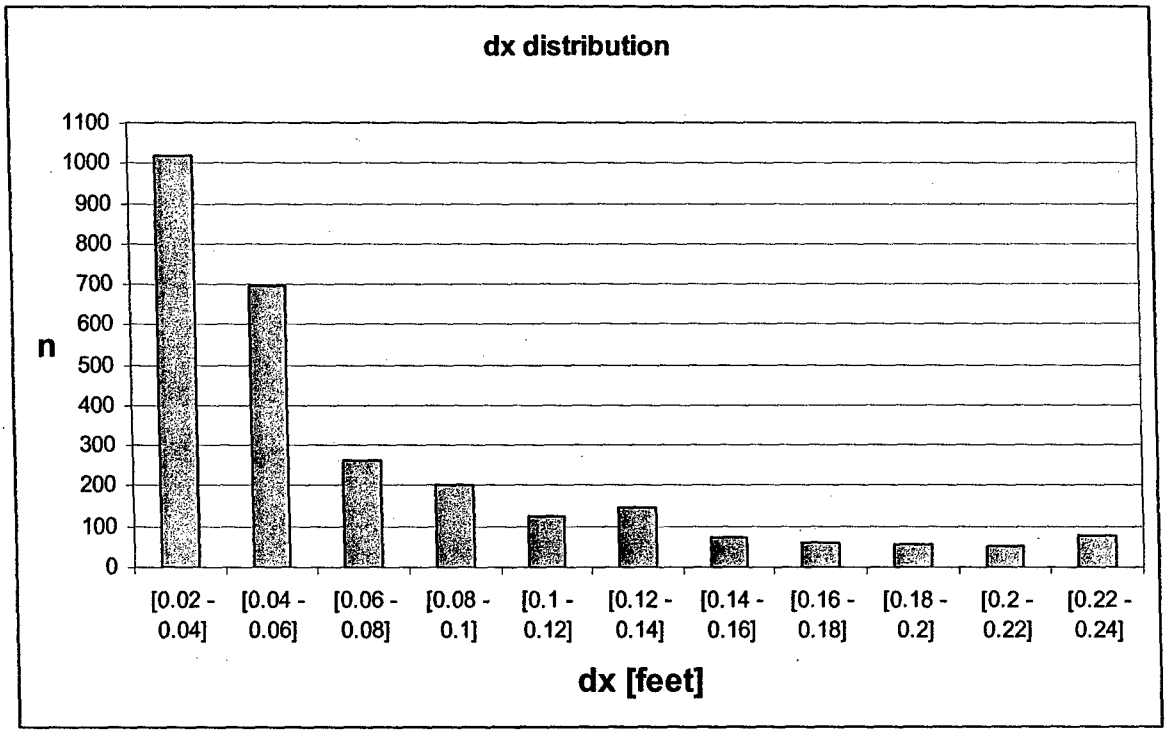


Figure 4b: *Distribution of dx (Δx as defined in Fig. 3c, d) for all decks of CVN-21. Most of the pairs of points have dx<0.02 ft (not shown); the displayed sample of the pairs constitutes about 6.75% of all pairs with dx<0.02 ft.*

As can be seen in Fig. 4a, there are hundreds of cases where the line segments are drawn off by 0.04-0.16 ft (approximately 0.5-2.0 inches) from their intended positions. Since in many cases a single line segment represents more than one bulkhead, this precision affects reconstruction of a significant fraction of compartments. Similarly, there are hundreds of cases where the distance between an endpoint of a bulkhead and another bulkhead forming a tee is between 0.5 and 2.0 inches.

The visibility of the precisions on the computer screen or a printout depends on the zoom factor used. Unless the drawing is aggressively zoomed in, Δx is below the resolution of the screen and thus, difficult to notice by the draftsman. As small as the inaccuracies are, they significantly change the procedure of automatic recognition of compartments. The precision affects the capability to (1) identify duplicated lines; (2) break lines; and (3) identify closed contours representing compartments.

To handle the inaccuracies, we have introduced a tolerance defined as a minimum distance between two points (corresponding endpoints of two bulkheads) to be considered as genuinely two distinct points. In other words, if the distance between two points is less than the tolerance, we assume that they coincide. We use the same tolerance (now representing the distance between a point and a line) to determine whether the line should be broken or not (c.f. Figs. 2 and 3c, d), or if two lines overlap or not (c.f. Fig. 3f).

In practice, if Δx is less than the tolerance, we "snap" the points, i.e., we change the coordinates of one point to match the other. In the case of a T, as in Fig. 3c,d, we move it to the

center of the line representing the bulkhead. As usual, this operation is done in the computer memory without modifications of the original drawing. Although this approach is very effective, it has two serious drawbacks.

The first drawback relates to the upper limit of the tolerance. The tolerance cannot be larger than the actual features in the drawings; otherwise, the features will be eliminated by this procedure. There are bulkheads that are about 1 inch long. This is usually the case when a door or an archway is located in that distance from a corner of the compartment. This is interpreted as another type of a drawing issue and will be discussed in more details in Section 6, "Reconstruction of Junctions". If the tolerance is greater than the length of such a bulkhead, our procedure of snapping will reduce its length to zero, and then eliminate it altogether as a zero-length feature. This creates "orphan" junctions, such as a door not attached to a bulkhead.

Another example of this tolerance issue involves a polyline representing a curved surface. It seems that these polylines are imported from other drawing software, most likely a solid modeling package. To approximate the curved shape of the bulkhead, the nodes of the polyline are spaced very tightly (in some cases there are polylines with 2000-3000 nodes 1/3 in. apart from each other), and this gets treated as another drawing problem. If the overall accuracy of the drawing is in the order of 1 in., there is no point of reproducing the shape of a curved bulkhead with a better resolution – most likely it is enough to tune the parameters of the import procedure. Conversely, if the shape of the curved bulkhead must be known with that precision, the rest of the drawing must be more accurate.

Some of the curved bulkheads could not be handled as they are drawn. Therefore, the original drawings were manually modified to increase the distance between the nodes of the polylines to be larger than the tolerance value.

The second drawback involves the snapping procedure which potentially introduces even more issues. The AutoCAD file does not provide any information on which of the two sets of coordinates are true, if any. Figure 5 shows schematically an example of this kind of ambiguity.

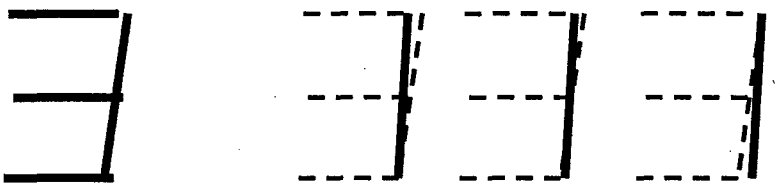


Figure 5: *For the shape on the left hand side where no nodes match but all are within our tolerance, what is the true intention of the draftsman? The CAD file provides no information to make a decision.*

In order to connect joints, the program makes a random choice (as good as any). It is a satisfactory solution for reconstruction of bulkheads, though it makes it more difficult to reconstruct decks and overheads, as is explained below.

Most of the problems with the aforementioned issues can be solved by utilizing AutoCAD capabilities to automatically snap to points of various types (e.g., end points, midpoints, and junctions). There are various modes of snapping which can be controlled to help

snap to the appropriate place on an object, and these are easily turned on/off using the "OSNAP" button in the tool bar.

4.5.3 Removing Duplicate Lines

In addition to zero-length lines, the CVN-21 CAD drawings are "contaminated" with redundant lines, that is, lines drawn twice (or in some cases, more times). They are referred to as line duplicates. Because of the problems in the drawings, several categories of duplicates are encountered, as schematically shown in Fig. 6.

Figure 6: *Examples of duplicate lines. The black areas shows overlap areas of the two line segments. From left to right: (1) two identical lines drawn exactly one on the top of the other; (2) a short line coinciding with a longer one; (3) partially overlapping parallel lines; (4) overlapping tilted lines. In each case* $\Delta x <$ *tolerance.*

Since the duplicates confuse the compartment reconstruction algorithms, we need to remove them. This operation is performed after parsing, so that the original drawings are not affected.

The duplicates are not visible on the computer screen because they lay one on the top of the other, and a tool can be provided for producing a list of them or for identifying them.

4,5,4 Breaking Lines

Before reconstructing the compartments, we need to address the issue of the identification bulkheads (c.f. Fig. 2). To this end, we are breaking the candidate bulkhead lines into segments corresponding to the actual bulkheads, as defined in Section 4. Conceptually, this is a very simple operation, as shown schematically in Fig. 7.

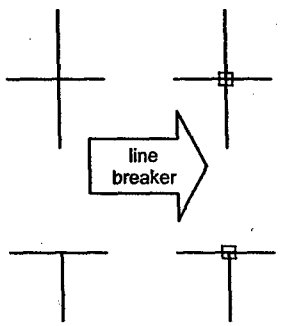


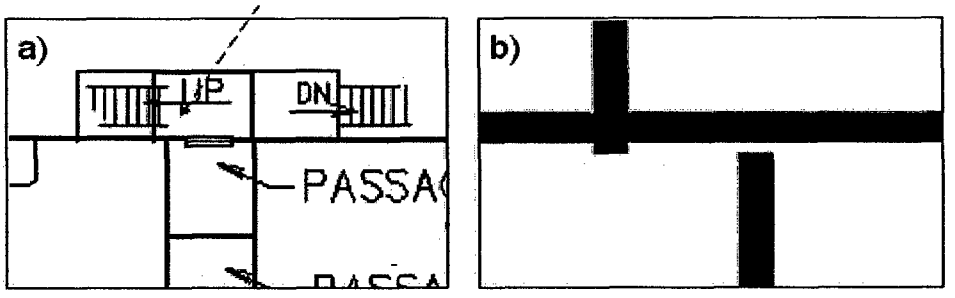
Figure 7: *Breaking lines by adding new endpoints.*

In general, there are two cases: either the line segments cross each other (upper part of Fig. 7) or they form a T (lower part of Fig. 7). To break the lines, the intersection between the line segments is found and the lines are split at that point. That is, for crossing segments, the line breaker procedure replaces two original segments by four line segments, each starting at the point marked by a small rectangle. For a T, the line breaker replaces two line segments by three.

Again this operation is done in the computer memory and does not affect the original drawings.

Unfortunately, in the case of CVN-21, this operation proves to be much more complex and ambiguous. An example in the actual CVN-21 drawing is shown in Fig. 8.

Figure 8: *An example of a problem when drawing bulkheads (end points off about [1.8"], which*



is more than our tolerance value). The difference between expected and actual position of points is too small to be seen (fig. a) unless aggressively zoomed-in (fig. b). The dashed-line arrow in Fig. a shows the location of the zoomed-in fragment shown in fig. b.

4.5.5. Removing Floating Line Segments

After snapping line segments' end points, removing duplicates, and breaking lines, it is verified that there are no floating line segments. A floating line segment is a segment with at least one endpoint which does not coincide with an endpoint of another line segment. Since our goal is identification of closed contours, existence of floating lines guarantees the failure of the compartment recognition algorithm. Therefore, we need to identify the floating lines and complete them by determining what the floating end or ends should really connect to, or by removing them, if necessary.

After snapping the endpoints, there are three sources of floating line segments. One is our conditioning procedure itself, in particular the line breaker. For example, when we deal with a pattern shown in Fig. 9 with Δx greater than the tolerance, the line breaker fails to recognize it as a T. Instead, it takes it as a cross pattern and consequently breaks both line segments. This results in generating an artificial line segment that obviously must be removed.

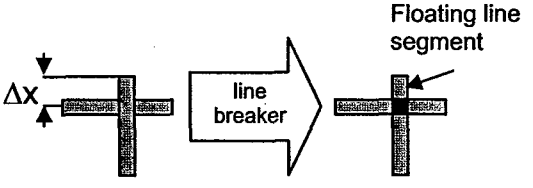


Figure 9: *A floating line generated as a byproduct of breaking lines.*

Another source of the floating line segments is intentional gaps in bulkheads coinciding with archways and windows (not that there is an inconsistency here, because another type of junction, namely doors, does not have corresponding gaps in bulkheads drawn in the CVN-21 CAD drawings). This problem is handled by filling in the gaps (in the computer memory, not in the actual drawings). The procedure for identification of junctions is described in section 6 below. Once the junction is identified, the corresponding gap is filled, as schematically shown in Fig. 10.

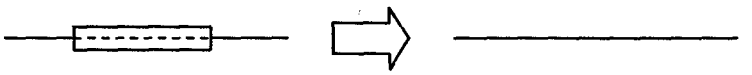


Figure 10: *Filling the gap representing an archway. The archway is drawn as a rectangle and a dashed line in a layer different than that used to draw bulkheads. Thus, in the bulkhead layer only two line segments (with a gap between them) are present. The gap is filled in by replacing both segments by a single long one.*

The most difficult sources of floating line segments are failures to identify bulkhead elements, as discussed in Section 4.2. The current procedure is to identify the floating segments and to visually inspect them. If the segment results from our conditioning procedure, it is simply deleted (those artifacts are being created in computer memory, and deleting them means deleting them from the computer memory). Otherwise, the bulkhead identification problem must be fixed (see Section 4.2), which often requires modification of the original CAD file.

4.6 Identification of Compartments

The resulting set of line segments that survived the conditioning and cleaning procedures are assigned a unique side identification and are used to identify closed contours which represent decks of compartments. Each compartment recognized in that way is assigned a unique compartment identification. The deck layout is extruded into the third dimension to form a compartment. To do that, we use the same algorithms (and actual code) that we developed for analyzing ex-USS Peterson. The only significant modification needed was the adding of the recognition of islands.

4.7 Identification of Bulkheads

The compartment identification procedure selects the line segments that represent bulkheads comprising a compartment. In fact, through this procedure, we identify sides of compartments and not bulkheads. A bulkhead is a surface that separates two compartments and has some properties, such as type of material and thickness. Also, the junctions, e.g. doors, are associated with bulkheads, and not sides, as they connect two compartments. Therefore, after the compartments are identified, it is necessary to "construct" bulkheads by creating a relationship between compartment sides. More specifically, pairs of matching (i.e., coinciding in space) compartment sides are identified, and labeled as pairs of bulkheads. Each bulkhead is then assigned a unique bulkhead identification. In addition, the bulkhead is identified by a pair of side identifications that form the bulkhead, and a pair of compartment identifications which are separated by this bulkhead. The identification of bulkheads is performed using the codes developed for ex-USS Peterson.

4.8 Reconstruction Of Deck And Overhead Segments

As mentioned in section 4.1, the compartment deck and overhead as recognized by the compartment recognition procedure (Section 4.6) does not have the fundamental property that they separate only two compartments. In general, the deck plans of two adjacent decks are different, and as a result, the deck of a compartment on one deck may match a group of

compartments on the deck below, not a one-to-one correspondence, as shown in Fig. 11. Therefore we need to split them into segments.

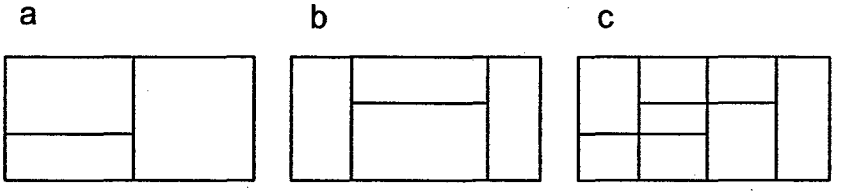


Figure 11: *Fragments of two adjacent decks (Figs. a and b) superimposed on top of each other (Fig. c). In such a case, the deck and the corresponding overhead of the other need to be split into segments.*

Conceptually, the procedure is simple. We superimpose deck plans of adjacent decks and repeat the compartment recognition procedure (c.f. Fig. 11). This results in identification of closed contours that represent floor and overhead segments. In other words, we need to repeat the procedures described in Sections 4.5 and 4.6: conditioning and recognition of closed contours.

In practice, the conditioning (that is, eliminating of the zero-length, duplicating and floating line segments) is more complicated than in the case of compartment recognition because of additional sources of uncertainty and ambiguity. Among them are:

1. The shape and size of different decks are genuinely different.
2. Most of the decks of CVN-21 are contained in different AutoCAD files (one deck per file), and we do not know how accurately the decks are positioned in the file local reference system (AutoCAD provides a support for co-positioning objects drawn in different files). The remaining (small) decks are drawn in a single file. This means that these decks are not positioned correctly with respect to the local coordinate system of the file. We manually split that file into many files in order to have one correctly positioned deck per file.
3. Because of the possible inaccuracy of the deck positioning and problems in drawing described in Section 4.5.2, we cannot tell if a small difference between bulkhead positions within our tolerance value comes from the actual differences in the ship's design or they result from inaccuracies in drawings. Accepting all the differences as the intended bulkheads position would result in splitting decks and overheads in a large number of very small segments. This is not desired; therefore, we made our best effort to avoid that.
4. As described in section 4.5, to reconstruct compartments, the line segments are conditioned which result in an arbitrary adjustment of the endpoint position within the tolerance. This potentially adds to the inaccuracies; since if we superimpose two decks, we face cases where the endpoints that are intended to coincide are moved apart by our snapping procedure beyond our tolerance. A natural approach would be to increase the tolerance in order to accommodate this error compounding effect. Unfortunately, we cannot do that, because increasing the tolerance would eliminate some features from the drawing. To overcome this problem, we examined each case separately and manually

modified the original drawings as needed. Identifying the problems and correcting them is a very laborious procedure, as schematically shown in Fig. 12.

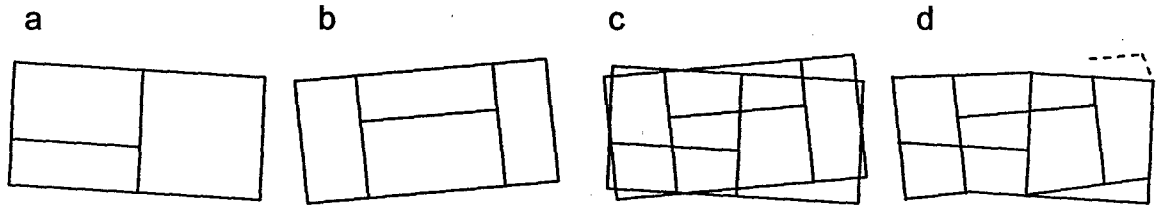


Figure 12: *Typical problem in the reconstruction of floor and overhead segments. If decks a and b are drawn inaccurately, the superposition of the decks might result in a complex pattern as shown in Fig. c (here exaggerated for the sake of clarity of this diagram). The line segment conditioning procedure would likely change the pattern to that shown in Fig. d by snapping points and eliminating duplicate line segments within the tolerances. However, the two line segments drawn with dashed lines in Fig. d would not be removed because the distances of their endpoints from the other line segments are larger than the tolerance. Because of these two floating lines, the recognition of the closed contours must fail. Note also that because of inaccuracies, the floor is split into 9 segments and not 8 as shown in Fig. 11.*

At least some of the complexity of deck and overhead segment recognition would be eliminated by replacing our current snapping procedure that randomly adjusts the position of endpoints within the tolerances. The new procedure would, for example, find the optimal location of the endpoints that maximizes overlap between bulkhead positions on the adjacent decks. Currently this procedure is done manually, case by case, as shown in Fig. 12.

4.9 Assigning Compartment Labels and Names

The final step in the compartment reconstruction procedure is associating the proper name and locator labels (deck-frame-position-user type) to the reconstructed compartments. These can be read from the CAD file. In DXF format, we have access to both the actual text of a label and its position in the local coordinate system (i.e., the text insertion point). Fortunately for us, in CVN-21, the labels are drawn in separate layers (compartment names in one layer and locator labels in another). Therefore, there is no need to parse the text to tell which is which (i.e., name or locator).

If the label is drawn inside the compartment, as shown in Fig. 13a, then the procedure is straightforward. To associate the label with a compartment, the list of compartments is searched to identify the compartment containing the text insertion point inside its floor perimeter. As shown in Fig 13b, not all compartment labels are positioned inside the compartment. This is understandable, since not all compartments are large enough to fit the labels. In general, we noticed a good effort of the draftsman to position the labels in such a way to maximize legibility of the drawings. This is a feature that we do not recommend to compromise.

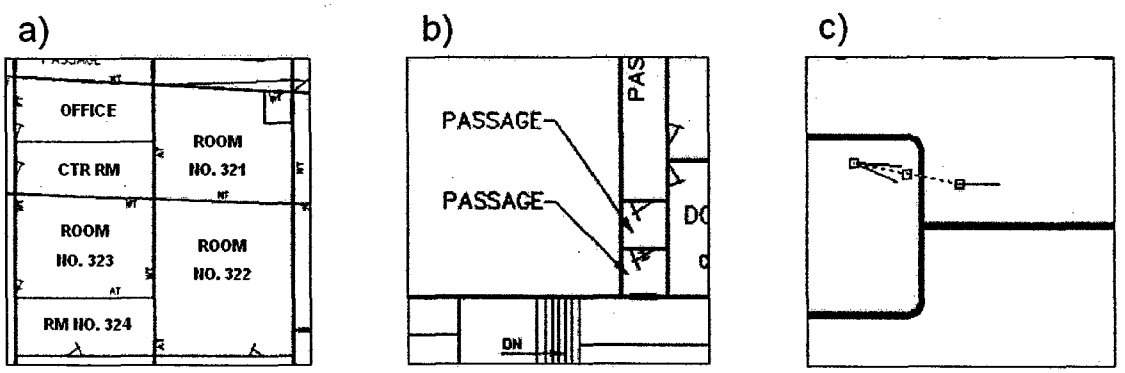


Figure 13: Compartment labels. In most cases they are drawn inside the compartments (a), but if there is not enough space, they are drawn outside with arrows pointing to them (b). Unfortunately, arrows are drawn inconsistently, in many cases as a set of separate lines segments (c), small rectangles show a selected line segment.

The labels that are drawn outside the compartments are visually associated with the compartment by drawing an arrow (Fig. 13b). In general, this is sufficient to automate processing of the labels. However, the way the arrows are drawn in the CVN-21 CAD drawings is inconsistent and difficult to process. Many of those arrows are not drawn as a single object of type `leader`, but as a collection of 4 lines. In order to reconstruct these arrows, we have to try all combinations of all lines in the layer (additionally, they are drawn with the same limited accuracy as the bulkheads, which adds to the complexity). Furthermore, even if an arrow is drawn as the `leader` object, the CVN-21 current practices do not take advantage of the capability of AutoCAD to associate the text with it. Instead, the text pointed to by the arrow is drawn as a separate AutoCAD object. This forces us to search all `text` or `Mtext` objects in the layer to find one that corresponds to the arrow. Arbitrary mixing of `text` or `Mtext` objects (`Mtext` is a multi-line text) is another example of inconsistencies in the CVN-21 drawings. A multi-line text is sometimes drawn as a `Mtext` and in other cases, each line is drawn as a separate `text` object, which makes the automatic processing more difficult. Given the time constraints to complete this project, the labels were manually moved to make their insertion points lay inside the compartments. For the long term, this is not a good solution.

Since we do not know the history of the AutoCAD files that we are processing, it is possible that the compartment labels were originally drawn as `leader` objects. A subsequent editing could have converted these leaders into the primitive elements, such as line segments and simple text objects, by use of the "explode" command, thus losing information important for the digital processing.

5. COMPARTMENT PROPERTIES

Once the compartments are reconstructed and properly labeled, it is straightforward to compute their properties which are required by the network fire model, such as volume, nominal extents and area of sides, as well as associate load types (there is one-to-one correspondence between the compartment use type and a load type defined in a database).

6. RECONSTRUCTION OF JUNCTIONS

A junction, in the network fire model terminology, is as an opening in a bulkhead or a deck that allows flow of air between compartments. In this project, we process the following types of junctions: doors, archways, portholes, hatches, hatchways, ventilation ducting, and scuttles.

We have developed algorithms for processing the junctions for our previous test case, ex-USS Peterson. Each junction type has been drawn as an AutoCAD block. Therefore, from the DXF representation of the CAD file, we were able to read the position of each junction (so-called block insertion point) and junction type (there is a one-to-one correspondence between the junction type and a block). Consequently, processing of junctions for the ex-USS Peterson was a straightforward procedure.

Unfortunately, the shipyard convention is not that way. The junctions are drawn as collections of lines, polylines and arches, as shown in Fig. 14, and not block references.

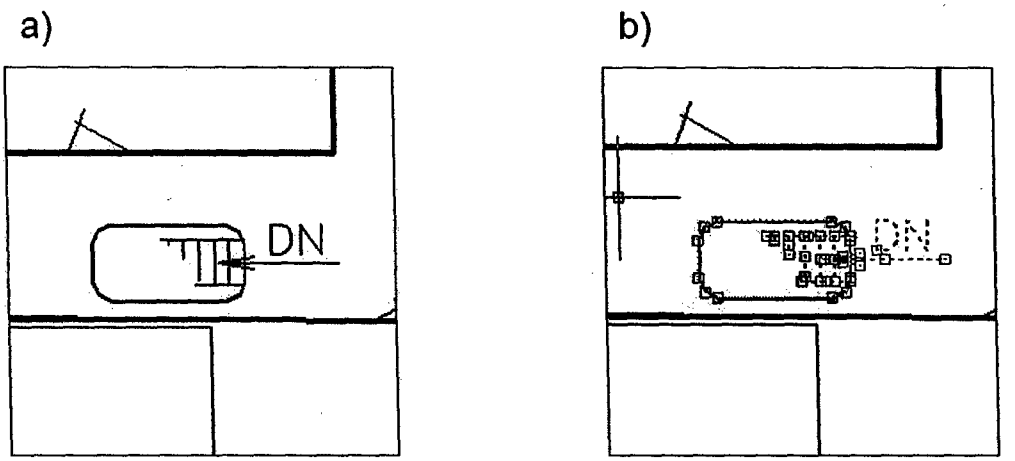


Figure 14: *Example of a hatch in CVN-21 CAD files. The hatch is drawn as a closed polyline with a ladder symbol and an arrow (a). This pattern has been laboriously drawn as independent graphical elements (b).*

This method of drawing a hatch is laborious (c.f. Fig. 14 and Fig. 15) and could be improved.

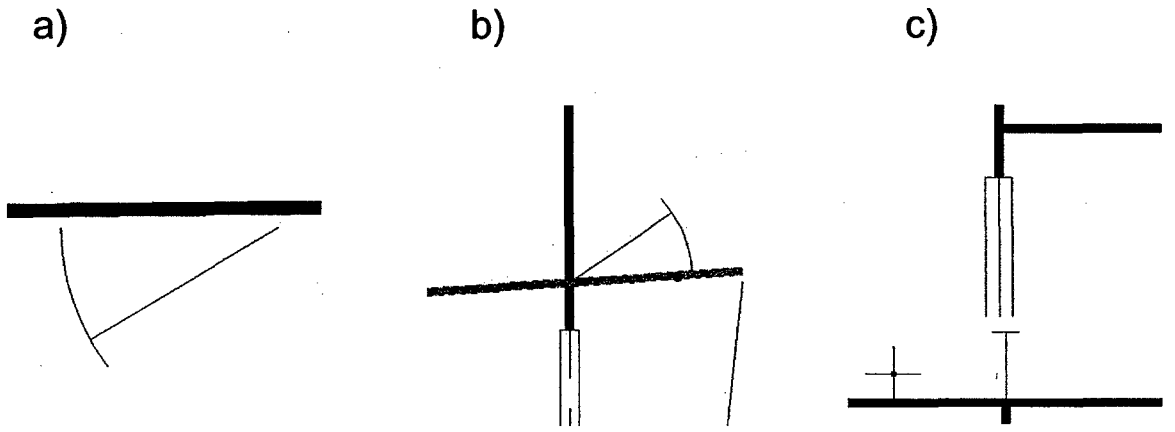


Figure 15: *Examples of inconsistently drawn junctions: (a) "floating" door, not attached to a bulkhead; (b) door "touching" two bulkheads; (c) archway*

The use of block definitions can significantly increase the accuracy, save time and effort, and improve conversion capabilities. For example, when the very first hatch is drawn, it can be saved as a block with multiple attributes and then recalled each time a new hatch is needed. Another method that is used to create a similar effect is by copying (or cut and pasting) objects. In our opinion, the block definitions method can be more efficiently used to update the drawings automatically, when required.

The blocks have yet another advantage. One can associate user-defined properties with a block. This is of critical importance for vertical junctions such as doors. The CVN-21 AutoCAD drawings are two-dimensional. There is no way of capturing the variable height of the doors or the presence and size of the sill, short of inserting an explanatory text. Taking into account the inconsistency of inserting the text (c.f. Section 4.9), a programmatic identification of door sizes is very difficult as opposed to a straightforward reading of the block properties.

Without blocks, the reconstruction of junctions becomes very complex. For example, a door is represented as a line and an arc. To determine the location of the door, we have to select the endpoint of the line that is opposite to that associated with the arc and find a bulkhead to which the endpoint belongs. If the line is drawn as in Fig 15a, we need to use tolerances. However, some doors are drawn very close to other bulkheads (see Fig. 15b), thus making the procedure ambiguous. Once we know the door insertion point (a "hinge"), we need to determine where the other end of the door is, if it is to the right or to the left of the hinge. To this end, we examine the endpoints of the arc. We can determine which arc belongs to which line by trying all combinations of lines and arcs in the layer containing the doors. Figure 15b shows another complication. The other end of the door is associated with a different line segment (i.e., a different bulkhead) than the hinge. To handle such cases, more elaborate algorithms are needed. All this could be avoided if the doors were drawn as blocks.

There are other issues that further add to the complexity of programmatic identification of junctions. For example, an archway on the computer screen or a printout looks like the one shown in Fig. 1. Many of them are drawn as a closed polyline (a rectangle) with a dashed line in

the middle connecting the bulkheads on both sides of the archway. For others, instead of a polyline, four line segments forming a rectangle are drawn, as shown in Fig 15c. Yet others are drawn without the dashed line in the middle. This makes it virtually impossible to define a unique signature of an archway for a programmatic recognition. We were not able to proceed without performing manual corrections to the draftsman drawings. The dashed line was used as the archway signature and manually added, whenever they were missing. The AutoCAD blocks prevent this type of problem from occurring.

The hatches were drawn in such a way (Fig. 14c) that the decision was made to redraw them manually as blocks rather than trying to develop algorithms to recognize them.

7. JUNCTION PROPERTIES

For the purpose of the FSSIM, the junctions are defined as vent opening flow paths between compartments and other compartments and between compartments and the surrounding ambient space. They may be horizontal as in a deck/overhead boundary (e.g., hatches and scuttles) or vertical as in a bulkhead/wall boundary (e.g., doors, portholes, and archways).

A junction is characterized by its location (a pair of compartment identifications that are connected through this junction, orientation and elevation), by its size (height, width, and/or area) and its hydraulic properties, such as the flow loss coefficient K and inertial length. These junction properties can be easily obtained, either directly from the database describing the ship after the reconstruction or derived from the database following the algorithms provided to us by the FSSIM authors from Hughes Associates, Inc.

8. CONCLUSIONS

In spite of the phenomenal rate of advancement in hardware and software, a computer cannot compete with the human eye and brain in recognizing and understanding a complex drawing. Therefore, in order to process the CAD files programmatically, the drawings must conform to certain rules that simplify the analysis. Certainly, there is a cost associated with converting the existing drawings to such standards, but there are also enormous benefits of digital processing that may outweigh the effort. Numerical modeling may emerge in the future as a critical element of the ship design process. Automatic conversion of CAD into digital geometries allows almost instantaneous utilization of numerical modeling to evaluate ship designs and design philosophies and to quickly arrive at an overall concept to meet required performance goals. As the ship concept is refined, simulations can be continued to evaluate ship vulnerability and to begin the process of defining ship operations.

Although we do not know the environment in which the CVN-21 drawings were generated, we think that one of the problems is that the drawings are created not from scratch but by modifying results of some previous efforts. This would explain inconsistencies in how conceptually similar elements are drawn. Also, we are aware that the drawings include elements imported from other tools that are used concurrently with AutoCAD (e.g., to design the hull).

To achieve the goal of making the CVN-21 drawings available for digital processing, the following suggestions are made:

(1) Use the tools already built into AutoCAD, such as automatic snapping, reconciling coordinate systems across different files representing different parts of the same ship, as well as blocks and attributes. Using blocks allows specifying properties and attributes of objects in the AutoCAD drawing. For example, the properties might describe the door types and heights, the existence and size of sills (otherwise difficult to represent in a two-dimensional drawing), thickness of bulkheads and materials used to construct them. Unfortunately, AutoCAD cannot recognize all unintentional mistakes such as putting an object in the wrong layer or using wrong line type to draw an element, but these practices will remedy some of the problems.

(2) In many cases, it is virtually impossible to see mistakes during creation, such as duplicate lines, zero-length lines, tiny misalignments and so forth. We can develop software tools that can detect these kinds of problems. Such tools would serve the following purposes, simultaneously. It would assist the draftsman in correcting unintentional mistakes, otherwise difficult to detect, and provide the capability for improving inconsistencies and controlling the quality of drawings.

(3) Another issue we noticed is the versioning problem. As the design process progresses, changes are introduced in the drawings. The changes need to be approved and occasionally rolled back. In many cases, it is desired to simultaneously maintain several versions of the design before a decision is made, and consequently, there is a need for comparing the competing versions visually or even better, by running numerical simulations. Finally, the updated versions must be shared between designers working on the same project. The current solution of placing drawing changes into a separate layer, while effective in some aspects, produces several drawbacks. One is that if an element of the drawing should be moved into a different position or removed altogether, it must be done in the layer where it was originally drawn. This makes it difficult to compare different versions and rolling back this category of change. Another problem is that corrections for a particular revision are drawn in a single layer, which makes automatic processing very difficult. Finally, there is no mechanism for controlling the use of the latest drawing revision by all parties.

The versioning is a known, and, to a large extent, a solved problem in the software development communities. These techniques could be applied to manage revisions of the AutoCAD drawings. For example, each layer of an AutoCAD drawing could be checked in separately into a repository, such as Concurrent Versioning System (CVS). This would allow modification of each layer individually which could be checked in as a new version. To see the complete design, all layers of the given revision could be checked out and programmatically reassembled as a complete AutoCAD file, or alternatively, different versions of the same layer could be combined into a single file to see the differences. Furthermore, using this technique,

corresponding layers from different files could be superimposed, one on top of another, thus giving the capability of comparing drawings of, e.g., different decks. As a result, a different version of the design could be unambiguously displayed, compared with other versions, or fed into post-processing software. The repository would be shared between users (draftsmen and designers), and this would solve the problem of consistency of the design and analysis. It is possible that this or a similar functionality is available from commercial "plugs-in" to AutoCAD.

Within this project, parts of the CVN-21 representing two partially overlapping test areas have been successfully reconstructed. The AutoCAD files have been converted into a digital representation of the ship and stored in a relational database. The digital representation of the ship has been used for generating three-dimensional visual model of the ship that serves as a Graphical User Interface (GUI) for the FSSIM, and is used for visualizations of the simulation results. Most importantly, we use the information stored in the database for the automatic generation of the input files for the fire model.

Most of the algorithms for the reconstruction came from the previous work involving the reconstruction of a small test area of ex-USS Peterson. Because of its size, the CVN-21 project proved to be far more complex, revealing some deficiencies of our software that needed to be analyzed and corrected. However, the most laborious part of this effort was dealing with drafting practices. The most serious problems, such as inconsistencies in positioning of the ship elements, were handled by developing complex algorithms. Similarly, additional procedures were developed for recognition of patterns representing doors, portholes and archways, instead of a straightforward processing of AutoCAD blocks, as was the case for ex-USS Peterson. Finally, to handle numerous exceptions and unnecessarily complicated patterns, the shipyard were manually corrected.

To take full advantage of numerical simulations as a design optimization tool, it is necessary to automate the process of converting the CAD drawings into the model's input files. This work demonstrates that it is feasible, provided the drawings are prepared with digital processing in mind and drafting practices follow certain rules making the drawings unambiguous. To achieve that, it is necessary to employ procedures and software tools to assess the conformance of the drawing to such standards. The expected result of making these changes would allow a rapid, programmatic extraction of the ship geometry to become the input for numerical simulations of various aspects of the ship design.

9. REFERENCES

[1] AutoCAD ® is a product of AutoDesk, Inc, home page http://www.autodesk.com

[2] J. Floyd, S. Hunt, F. Williams, and P. Tatem, "Fire and Smoke Simulator (FSSIM) Version 1 – Theory manual", NRL/MR/6180-04-8765, 2004.

[3] J. Floyd, S. Hunt, P. Tatem, and F. Williams, "Fire and Smoke Simulator (FSSIM) Version 1 - User Guide", Naval NRL/MR/6180-04-8806, 2004.

[4] DXF specification can be obtained from many internet repositories, for example, from http://www.faqs.org/faqs/graphics/fileformats-faq/part3/section-45.html

[5] T. Haupt, D. Shulga, B. Sura, S. Durvasula, P. Tatem, and F. Williams, "Simulation Environment for Onboard Fire Network Model Version 1.0 – Theory Manual, NRL/MR/6180-04-8800, 2004.